H

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/784,505	02/15/2001	James Andrew Adams	343355600029	3316

7590 08/13/2004

Jones, Day, Reavis & Pogue
North Point
901 Lakeside Avenue
Cleveland, OH 44114

EXAMINER	
ROCHE, TRENTON J	

ART UNIT	PAPER NUMBER
2124	

DATE MAILED: 08/13/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/784,505	ADAMS, JAMES ANDREW	
	Examiner	Art Unit	
	Trent J Roche	2124	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *13 May 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-28* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-28* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *15 February 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This office action is responsive to Amendment A filed 13 May 2004.

2. Per applicant's request, amended claims 1 and 18 have been entered. Claims 1-28 are now pending.

3. Claims 1-28 have been examined.

Claim Rejections - 35 USC § 102

4. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.

5. Claims 1-7, 9-24 and 26-28 are rejected under 35 U.S.C. 102(a) as being anticipated by "Dynamic Class Loading in the Java™ Virtual Machine" by Liang et al (hereafter referred to as Liang).

Regarding claim 1:

Liang teaches:

- a computer-implemented method for loading an object class into computer memory for access by an object-oriented application ("Class files are produced by Java compilers, and can be loaded into any Java virtual machine. A class file does not have to be stored in an actual file; it could be stored in a memory buffer..." in section 2, page 37.)

- loading into the computer memory a first object class, wherein a classload request for the first object class traverses from a first classloader to a second classloader ("A class loader L_1 can ask another loader L_2 to load a class C on its behalf" in section 2.2, page 38)
- wherein a class loading operation accesses the second classloader ("A class loader L_1 can ask another loader L_2 to load a class C on its behalf" in section 2.2, page 38. Since L_2 is asked to load class C, it is accessed.)
- creating a destructible classloader that contains a second object class ("Subclasses of ClassLoader can override the definition of loadClass, thus providing a user-defined loading policy. Here is a class loader that looks up classes in a given directory...class MyClassLoader extends ClassLoader {..." in section 2.3, page 38)
- after creating the destructible classloader, the classload request traverses from the first classloader to the destructible classloader and then to the second classloader ("if findLoadedClass returns null, the class has not yet been loaded. We then delegate to the system class loader by calling findSystemClass. If the class we are trying to load is not a system class, we call a helper method getClassData to read in the class file" in section 2.3, page 38)
- wherein after creating the destructible classloader, the destructible classloader is accessed when the class loading operation with respect to the second classloader is utilized ("allows a new version of the Service class to be dynamically loaded, replacing the existing service object...Further requests will be redirected to the new object referenced to by service..." in section 3.1, page 39)
- wherein the destructible classloader attempts to load into the computer memory the second object class if the second classloader fails to load the second object class ("If the class we are

trying to load is not a system class, we call a helper method getClassData to read in the class file" in section 2.3, page 38)

as claimed.

Regarding claim 2:

The rejection of claim 1 is incorporated, and further, Liang discloses the destructible classloader containing a variation of the classes specified in the first object class as claimed ("a class type is uniquely determined by the combination of the class name and class loader...a class named C loaded in applet 1 is considered a different type from a class named C in applet 2" in section 2.2, page 38)

Regarding claim 3:

The rejection of claim 1 is incorporated, and further, Liang discloses unloading the destructible classloader, creating a second destructible classloader that contains a third object class, wherein after creating the second destructible class, a classload request for the third object class traverses from the first classloader to the second destructible classloader and then to the second classloader, and the second destructible classloader attempts to load the third object class as claimed (Note the rejection of claim 1 regarding the placement of a first destructible classloader. Further, "this ability translates to reloading a subset of the classes already loaded in a running virtual machine...new classes are loaded by a separate loader...a Server class can dynamically redirect the service requests to a new version of the Service class. The key technique is to load the server class, old service class, and new service class into separate class loaders" in section 3.1, page 39. The new server class classloader would replace the old server class classloader in the loading hierarchy. Finally, "the two Service classes may coexist for a while, until all uses of the old class are complete, all references to the old

class are dropped, and the old class is unloaded" in section 3.1, page 40. This would unload the classloader as well, as is indicated by "Classes are unloaded when their defining loader is garbage-collected" in section 2.2, page 38)

Regarding claim 4:

The rejection of claim 3 is incorporated, and further, Liang discloses the second destructible classloader containing a variation of the classes specified in the second object class as claimed ("Figure 3 illustrates how a Server class can dynamically redirect the service requests to a new version of the Service class" in section 3.1, page 39)

Regarding claim 5:

The rejection of claim 1 is incorporated, and further, Liang discloses an object-oriented development environment as claimed ("Java Development Kit..." in section 1, page 36)

Regarding claim 6:

The rejection of claim 5 is incorporated, and further, Liang discloses the second object class being made accessible while the development application is operating as claimed ("The upgrade must not require the application to shut down and restart. On the Java platform, this ability translates to reloading a subset of the classes already loaded in a running virtual machine" in section 3.1, page 39)

Regarding claim 7:

The rejection of claim 7 is incorporated, and further, Liang discloses the object-oriented application operating substantially throughout a twenty-four hour period as claimed ("It is often desirable to

upgrade software components in a long-running application such as a server" in section 3.1, page 39. Long-running for a server application is interpreted to extend over several days, thus operating substantially throughout the twenty-four hour period of each day.)

Regarding claim 9:

The rejection of claim 1 is incorporated, and further, Liang discloses a first classloader originating the classload request as claimed ("If findLoadedClass returns null, the class has not yet been loaded" in section 2.3, page 38)

Regarding claim 10:

The rejection of claim 9 is incorporated, and further, Liang discloses the second classloader being a system classloader ("We then delegate to the system class loader..." in section 2.3, page 38)

Regarding claim 11:

The rejection of claim 10 is incorporated, and further, Liang discloses a Java-based environment as claimed (Note the rejection regarding claim 5)

Regarding claim 12:

The rejection of claim 1 is incorporated, and further, Liang discloses requests traveling to the destructible classloader as claimed ("If the class we are trying to load is not a system class, we call a helper method getClassData to read in the class file" in section 2.3, page 38. Further, the process of a classload request traveling from a user classloader to a system classloader to an extensions classloader to a bootstrap classloader is inherent in the system disclosed by Liang. "Understanding

Extension Class Loading" discloses that such a process is built into the Java™ platform. "The extension framework makes use of the new class loading mechanism in version 1.2 of the Java™ platform. When the runtime environment needs to load a new class, it looks for the class in the following locations, in order: 1. bootstrap classes...2. installed extensions...3. the class path – classes on paths specified by the system property java.class.path..." on page 1 of "Understanding Extensions Class Loading.")

Regarding claim 13:

The rejection of claim 1 is incorporated, and further, Liang discloses manipulating the method such that the method returns the destructible classloader as claimed ("if the class is not a system class, we call a helper method getClassData to read in the class file" in section 2.3, page 38)

Regarding claim 14:

The rejection of claim 13 is incorporated, and further, Liang discloses the destructible classloader being a child as claimed ("all user-defined class loader classes are subclasses of ClassLoader" in section 2.3, page 38)

Regarding claim 15:

The rejection of claim 14 is incorporated, and further, note rejection regarding claim 10.

Regarding claim 16:

Liang teaches:

- a computer-implemented system for loading object classes into computer memory for access by an object-oriented application ("Class files are produced by Java compilers, and can be loaded into any Java virtual machine. A class file does not have to be stored in an actual file; it could be stored in a memory buffer..." in section 2, page 37.)
- a pre-existing class loading hierarchy that specifies parent-child relationships of classloaders (Note Figure 2 and the corresponding discussion in section 2.2)
- a class loading operation that provides one of the classloaders in the pre-existing class loading hierarchy when the class loading operation is called, wherein the provided classloader is used to make accessible to the application an object class ("The Java virtual machine uses class loaders to load class files and create class objects" in section 2.1, page 37)
- a classloader switching module that inserts a first destructible classloader into the pre-existing class loading hierarchy by causing the class loading operation to provide the first destructible classloader when the class loading operation is called ("A class loader L_1 can ask another loader L_2 to load a class C on its behalf. In such a case, L_1 delegates C to L_2" in section 2.2, page 38)
- after the first destructible classloader is inserted into the pre-existing class loading hierarchy, the first destructible classloader makes accessible for loading into the computer memory a first object class ("If the class we are trying to load is not a system class, we call a helper method getClassData to read in the class file" in section 2.3, page 38)

as claimed.

Regarding claim 17:

The rejection of claim 16 is incorporated, and further, Liang discloses the first destructible classloader containing the first object class as claimed (Note Figure 3. The old service class, which is associated with a first classloader, contains the first object class, which is later replaced.)

Regarding claim 18:

The rejection of claim 16 is incorporated, and further, Liang discloses inserting the first destructible classloader into the pre-existing class loading hierarchy, wherein the first destructible classloader is returned as the parent of a user classloader instead of a system classloader, wherein the system classloader is the parent of the first destructible classloader as claimed ("Classes are loaded on demand. Class loading is delayed as long as possible..." in section 1, page 36. The user-defined classes and their associated classloaders would not be loaded until a request is made. Further, note Figure 3, wherein there is "a new version of the Service class to be dynamically loaded, replacing the existing service object...Further requests will be redirected to the new object referenced to by service..." in section 3.1, page 39. The old Service user-defined class is replaced by the new Service class, and the newly created Service class is accessed rather than the old class. Further, all classes have the system class as a parent, as shown in section 2.4, page 38, wherein "Instances of the MyClassLoader class delegate the loading...to the system loader.")

Regarding claim 19:

The rejection of claim 16 is incorporated, and further, Liang discloses unloading the first destructible classloader and loading the second destructible classloader as claimed (Note Figure 3 and the corresponding discussion in section 3.)

Regarding claim 20:

The rejection of claim 19 is incorporated, and further, Liang discloses the second object class including classes that were not in the first object class as claimed ("It is often desirable to upgrade software components..." in section 3.1, page 38. An upgrade will include information not in the prior version.)

Regarding claim 21:

The rejection of claim 19 is incorporated, and further, Liang discloses the second object class including classes that are a variation of at least one class as claimed ("It is often desirable to upgrade software components..." in section 3.1, page 38)

Regarding claim 22:

The rejection of claim 16 is incorporated, and further, note rejection regarding claim 5.

Regarding claim 23:

The rejection of claim 22 is incorporated, and further, note rejection regarding claim 6.

Regarding claim 24:

The rejection of claim 16 is incorporated, and further, note rejection regarding claim 7.

Regarding claim 26:

The rejection of claim 16 is incorporated, and further, Liang discloses operating in a Java environment, the pre-existing class loading hierarchy including a system classloader, wherein the

switching module manipulates the class loading operation such that the class loading operation returns the first destructible classloader instead of the system classloader (Note rejections regarding claims 11 and 13)

Regarding claim 27:

The rejection of claim 26 is incorporated, and further, note rejection regarding claim 14.

Regarding claim 28:

The rejection of claim 27 is incorporated, and further, not rejection regarding claim 11.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 8 and 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Dynamic Class Loading in the Java™ Virtual Machine" by Liang et al (hereafter referred to as Liang), in view of U.S. Patent 6,675,381 to Yamaguchi.

Regarding claim 8:

The rejection of claim 1 is incorporated, and further, Liang discloses the object-oriented application consisting of a server ("in a long-running application such as a server" in section 3.1, page 39) Liang

does not explicitly disclose a web server. Yamaguchi discloses an analogous dynamic loading system a web server ("in case that the application is the Web server..." in col. 8 line 8) It would have been obvious to one of ordinary skill in the art at the time the invention was made to use the web server of Yamaguchi as the server disclosed by Liang, as this would enable a user utilizing the system disclosed by Liang to distribute objects over the Internet.

Regarding claim 25:

The rejection of claim 16 is incorporated, and further, note the rejection regarding claim 8.

Response to Arguments

8. Applicant's arguments filed 13 May 2004 have been fully considered but they are not persuasive.

Per claims 1, 16 and 18:

The applicant states that Liang et al do not disclose, teach or suggest modifying the classloader hierarchy such that a different classloader is returned instead of the "normal" or pre-existing classloader (e.g., a system classloader). However, as indicated in the rejections stated above, Liang et al allows a user to define their own classloaders, further shown in section 2.3, page 38, wherein "Subclasses of ClassLoader can override the definition of loadClass, thus providing a user-defined loading policy." The user-defined classloader is not a system or pre-existing classloader, and the user-defined classloader is returned for all user-defined classes. Consequently, Liang et al does allow a user to return a different classloader rather than a "normal" or pre-existing classloader. For these reasons, the rejections of claims 1, 16 and 18 are proper and maintained.

Per claims 8 and 25:

As indicated above, the rejections of claims 1 and 16 are maintained. Further, the Applicant fails to show that the reasons to combine and motivations concerning the rejections of claims 8 and 25 are improper, and as such, the rejections of claims 8 and 25 are maintained.

Conclusion

9. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL.** See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

10. A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Trent J Roche whose telephone number is (703)305-4627. The examiner can normally be reached on Monday - Friday, 9:00 am - 5:30 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (703)305-9662. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Trent J Roche
Examiner
Art Unit 2124

TJR

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100